ISEL
Departamento de Engenharia Eletrónica e Telecomunicações e de Computadores

Licenciatura em Engenharia Informática e de Computadores

4º Trabalho Prático

Trabalho realizado por:

Nome: António Paulino Nº 50512

Nome: Bernardo Pereira Nº 50493

Turma: LEIC23D

Docente: João Patriarca

Arquitetura de Computadores

2022 / 2023 verão

16 de Junho de 2023

Índice

1. Descrição do Sistema [1]

Foi desenvolvido um sistema embebido baseado no processador P16 para medir o tempo de reação simples [2] de pessoas. Para tal, o sistema conta o tempo que medeia entre a ativação de um estímulo visual e a resposta do utilizador. O resultado apresentado pelo sistema consiste na diferença, em milissegundos, entre o tempo medido e o tempo de reação médio de um ser humano a um estímulo visual simples, estabelecido em, aproximadamente, 200 milissegundos [2].

2. Arquitetura do sistema. [1]

O protótipo foi desenvolvido recorrendo às placas SDP16 [3] e ATB e ao circuito Pico Timer/Counter (pTC) [4], conforme ilustrado na Figura 1.

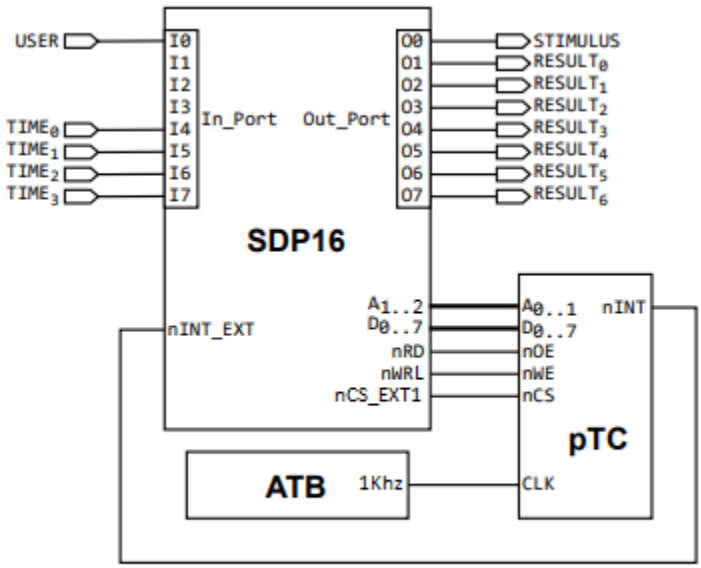

Figura 1: Diagrama de blocos do sistema implementado

Nesta implementação, o interruptor zero do DIP-switch 'SW1' instalado na placa SDP16 é utilizado para definir o valor da entrada USER, que estabelece a interação com o utilizador para iniciar os testes e recolher as respostas. Já o tempo de estímulo visual para a realização do teste é definido usando os interruptores 4 a 7 do DIP-switch 'SW1', que estabelecem o valor da entrada TIME. Os quatro bits desta entrada especificam um valor entre um segundo e 10 segundos.

Os oito bits do porto de saída instalado na placa SDP16 são utilizadas para mostrar várias informações ao utilizador, de forma individual ou combinada. O bit zero está associado à saída STIMULUS que é responsável pelo controlo do estímulo visual: ativo, quando STIMULUS='1', ou inativo, quando STIMULUS='0'. A saída RESULT está associada aos bits um a sete do porto de saída e apresenta o resultado do teste, um número inteiro com sinal compreendido na gama ±63. O valor −64 é utilizado para representar valores fora desta gama. Estes oito bits também são utilizados, em conjunto, para informar que o sistema está ligado e apto para funcionar.

Finalmente, o circuito pTC serve de suporte à realização das bases de tempo necessárias ao funcionamento do sistema. O sinal de relógio aplicado a este circuito é obtido do oscilador ('OSCILLATOR') disponível na placa ATB.

3. Especificação do funcionamento do sistema [1]

O sistema cumpre o seguinte modo de funcionamento:

1. No arranque do sistema deve garantir-se que a saída STIMULUS fica ativa, a saída RESULT apresenta o valor 0x7F e a entrada USER está inativa (USER='0').

2. Após esta etapa de iniciação, o estabelecimento do valor lógico '1' na entrada USER espoleta a realização de um novo teste de medição de tempo de reação, o que resulta na afixação do valor zero na saída RESULT e na defnição do tempo de estímulo para o teste.

3. Na primeira etapa do teste, a saída STIMULUS deve manter-se ativa durante o tempo de estímulo. Caso a entrada USER seja desativada antes de se cumprir a totalidade do tempo de estímulo, o teste é abortado e o sistema retorna à etapa de preparação descrita no ponto 1.

4. Na etapa seguinte do teste, a saída STIMULUS é desativada e o sistema fica a aguardar pela desativação da entrada USER, contabilizando o tempo que medeia entre ambos os eventos.

5. Na última etapa do teste, o resultado do teste é afixado na saída RESULT por cinco segundos. Decorrido esse tempo, o sistema retorna à etapa de preparação descrita no ponto 1.

4. Respostas às questões colocadas no enunciado

1. Apresente a solução adotada para ligar o circuito pTC à placa SDP16.

Para as ligações entre o circuito pTC e a placa SDP16, foi implementada a solução ilustrada na figura 1.

Foram ligados aos bits A0..A1 do pTC os bits A1..A2 da placa SDP16. Desta forma, os endereços da placa SDP16 utilizados para aceder os registos do pTC são pares (bit A0 da placa SDP16 fica sempre a 0), o que permite a ligação do sinal nWRL da placa ao circuito, já que para acessos de escrita de byte em endereços pares é utilizado o sinal nWRL.

Foram ligados os bits D0..D7 da placa SDP16 ao barramento de dados do circuito. Como foi referido, é feita escrita de byte em endereços pares e é utilizado o sinal nWRL. Então, foram ligados os bits da parte baixa do barramento de dados da placa ao circuito.

Finalmente, foi ligado o sinal nCS_EXT1 da placa ao sinal CS do circuito, o que permite que o circuito seja acessível na gama de endereços 0xFF40 a 0xFF7F.

2. Explique os cálculos realizados para determinar as temporizações envolvidas neste trabalho.

A temporização para o circuito pTC escolhida nesta implementação foi de 1KHz, ou 1 ciclo a cada milissegundo, sendo que foi utilizado um módulo de contagem de 10 milissegundos (10 ciclos). Isto significa que a cada 10 milissegundos é gerado um pedido de interrupção. O processador P16 tem uma frequência de 50KHz, ou um ciclo a cada 0.02 milissegundos. Tendo em conta que cada ciclo máquina dura 3 ciclos de relógio e uma instrução pode ter duração de 1 ciclo máquina ou 2 ciclos máquina quando envolve acessos a memória, uma instrução sem acesso a memória tem duração de 0,06 milissegundos e com acesso a memória 0,12 milissegundos. Tendo estas temporizações em conta, podem ser executadas $10 \div 0{,}06 = 166$ instruções sem acesso a memória ou $10 \div 0{,}12 = 83$ instruções com acesso a memória antes de ser gerado um pedido de interrupção, o que permite a execução do programa principal. A rotina de atendimento a pedidos de interrupção tem duas instruções sem acesso a memória, e 10 instruções com acesso a memória. Se a temporização não permitisse que fossem executadas mais instruções do que as que se encontram na rotina de interrupção, o sistema ficaria a atender pedidos de interrupção em loop infinito, já que seriam gerados pedidos mais rapidamente do que atendidos.

3. Indique, justificando, a latência máxima do sistema no atendimento dos pedidos de interrupção gerados pelo circuito pTC.

Tendo em conta que o processador avalia o estado do pedido de interrupção após a execução de cada instrução, a latência máxima será definida pelo tempo de execução dessa instrução. Para a maior latência possível, seria executada uma instrução com acesso a memória, que tem duração de 0,12 milissegundos (2 ciclos máquina), e depois seria atendido o pedido de interrupção. Então, a latência máxima no atendimento de pedidos de interrupção é de 0.12 millissegundos.

4. Indique, justificando, quanto tempo demora, no pior caso, a execução da rotina utilizada para o atendimento da interrupção externa.

Como foi referido na resposta 2, a rotina de atendimento a pedidos de interrupção tem duas instruções sem acesso a memória, e 10 instruções com acessos a memória. Tendo isto em conta, o tempo de execução da rotina é dado por $0.06\,ms \times 2 + 0.12\,ms \times 10 = 1.32\,ms$.

5. Conclusões

Neste trabalho, foi implementado um sistema embebido desenvolvido com base no processador P16 para medir o tempo de reação simples de pessoas. O sistema foi implementado utilizando as placas SDP16 e ATB, juntamente com o circuito Pico Timer/Counter (pTC).

O sistema permite a interação com o utilizador através de interruptores e exibe informação com recurso ao porto de saída da placa SDP16. O circuito pTC fornece as temporizações necessárias para o funcionamento do sistema, utilizando o sinal de relógio de 1KHz disponível na placa ATB.

O funcionamento do sistema foi separado por etapas, desde a inicialização até a apresentação dos resultados do teste. Estas etapas foram implementadas de acordo com uma máquina de estados em software assembly.

Na implementação, foram feitas as devidas ligações entre o circuito pTC e a placa SDP16, garantindo a comunicação adequada entre os componentes. Além disso, a temporização foi ajustada para permitir a execução do programa principal e o atendimento de interrupções.

Em conclusão, o sistema desenvolvido mostrou-se capaz de medir o tempo de reação simples de pessoas. A arquitetura implementada, juntamente com as configurações adequadas do circuito pTC e a comunicação correta com a placa SDP16, permitiram o funcionamento adequado do sistema.

6. Referências

[1]. Trabalho Prático 4 - Medição de Tempo de Reação. ISEL IPL, Lisboa, Portugal, maio 2023. (Acedido em 16-6-2023).

[2]. Kosinski, Robert J: A literature review on reaction time. Clemson University, 10(1):337 344, 2008.

[3]. Paraíso, José e Tiago Dias: Manual de Utilização da Placa de Desenvolvimento SDP16. ISEL IPL, Lisboa, Portugal, março 2023. (Acedido em 16-06-2023).

[4]. Dias, Tiago: Pico Timer/Counter (pTC) Product Datasheet. ISEL IPL, Lisboa, Portugal, v1.1.2 edição, junho 2021. (Acedido em 16-06-2023).

7. Listagem do programa

```
P16 assembler v1.4.0 (Mar  6 2023)  e:\Coisas uni\AC\lab05\tp4.lst      Fri Jun 16
15:12:53 2023

Sections
Index   Name            Address   Size
0       startup         0000      000E 14
1       .text           000E      0164 356
2       .data           0172      0020 32
3       .bss            0192      0004 4
4       .stack          0196      0040 64

Symbols
Name                    Type      Value       Section
_start                  LABEL     0004 4      startup
asm                     LABEL     0020 32     .text
AVG_TIME                ABSOLUTE  0014 20     startup
check_user              LABEL     009A 154    .text
CPSR_BIT_I              ABSOLUTE  0010 16     startup
delay                   LABEL     00CC 204    .text
delay_end               LABEL     00E0 224    .text
delay_loop              LABEL     00D8 216    .text
elapsed_return_false    LABEL     00B4 180    .text
get_res                 LABEL     00C8 200    .text
get_sleep_time          LABEL     00BC 188    .text
get_time_value          LABEL     00C4 196    .text
goto_init               LABEL     0064 100    .text
goto_wait_user_res      LABEL     0060 96     .text
inport_addr             LABEL     0114 276    .text
INPORT_ADDRESS          ABSOLUTE  FF80 65408  startup
inport_read             LABEL     0106 262    .text
inport_read_mask        LABEL     010C 268    .text
is_elapsed              LABEL     00A2 162    .text
is_elapsed_end          LABEL     00B6 182    .text
isr                     LABEL     00EC 236    .text
isr_addr                LABEL     000C 12     startup
main                    LABEL     000E 14     .text
main_addr               LABEL     000A 10     startup
MAX_TIME                ABSOLUTE  0040 64     startup
MIN_TIME                ABSOLUTE  FFC1 65473  startup
MIN_TIME_ADDR           LABEL     00B8 184    .text
OUT_OF_RANGE            ABSOLUTE  FFC0 65472  startup
out_of_range            LABEL     008A 138    .text
OUT_OF_RANGE_ADDR       LABEL     00BA 186    .text
outport_addr            LABEL     0146 326    .text
OUTPORT_ADDRESS         ABSOLUTE  FFC0 65472  startup
outport_clear_bits      LABEL     0124 292    .text
outport_img             LABEL     0192 402    .bss
outport_img_addr        LABEL     013E 318    .text
outport_init            LABEL     0134 308    .text
OUTPORT_INIT_VAL        ABSOLUTE  0000 0      startup
outport_set_bits        LABEL     0116 278    .text
outport_write           LABEL     0140 320    .text
PTC_ADDR                LABEL     0170 368    .text
PTC_ADDRESS             ABSOLUTE  FF40 65344  startup
PTC_CMD_START           ABSOLUTE  0000 0      startup
PTC_CMD_STOP            ABSOLUTE  0001 1      startup
ptc_get_value           LABEL     0158 344    .text
ptc_init                LABEL     015E 350    .text
```

```
ptc_start               LABEL     0148 328    .text
ptc_stop                LABEL     0150 336    .text
PTC_TC                  ABSOLUTE  0004 4      startup
PTC_TCR                 ABSOLUTE  0000 0      startup
PTC_TIR                 ABSOLUTE  0006 6      startup
PTC_TMR                 ABSOLUTE  0002 2      startup
RES_DISPLAY_TIME        ABSOLUTE  0005 5      startup
RESULT_MASK             ABSOLUTE  00FE 254    startup
show_res                LABEL     008C 140    .text
SLEEP_0                 ABSOLUTE  0032 50     startup
SLEEP_1                 ABSOLUTE  0064 100    startup
SLEEP_10                ABSOLUTE  03E8 1000   startup
SLEEP_11                ABSOLUTE  0226 550    startup
SLEEP_12                ABSOLUTE  028A 650    startup
SLEEP_13                ABSOLUTE  02EE 750    startup
SLEEP_14                ABSOLUTE  0352 850    startup
SLEEP_15                ABSOLUTE  03B6 950    startup
SLEEP_2                 ABSOLUTE  00C8 200    startup
SLEEP_3                 ABSOLUTE  012C 300    startup
SLEEP_4                 ABSOLUTE  0190 400    startup
SLEEP_5                 ABSOLUTE  01F4 500    startup
SLEEP_6                 ABSOLUTE  0258 600    startup
SLEEP_7                 ABSOLUTE  02BC 700    startup
SLEEP_8                 ABSOLUTE  0320 800    startup
SLEEP_9                 ABSOLUTE  0384 900    startup
sleep_ADDR              LABEL     0104 260    .text
sleep_times             LABEL     0172 370    .data
STACK_SIZE              ABSOLUTE  0040 64     startup
state_standby           LABEL     0030 48     .text
STATE_STANDBY           ABSOLUTE  0000 0      startup
state_wait_stimulus     LABEL     003C 60     .text
STATE_WAIT_STIMULUS     ABSOLUTE  0001 1      startup
state_wait_user_res     LABEL     0068 104    .text
STATE_WAIT_USER_RES     ABSOLUTE  0002 2      startup
states                  LABEL     002A 42     .text
STIMULUS_MASK           ABSOLUTE  0001 1      startup
sysclk                  LABEL     0194 404    .bss
sysclk_ADDR             LABEL     0102 258    .text
SYSCLK_FREQ             ABSOLUTE  0009 9      startup
sysclk_get_ticks        LABEL     00E6 230    .text
SYSTEM_INIT             ABSOLUTE  00FF 255    startup
TIME_MASK               ABSOLUTE  00F0 240    startup
tos                     LABEL     01D6 470    .stack
tos_addr                LABEL     0008 8      startup
USER_MASK               ABSOLUTE  0001 1      startup
user_react              LABEL     0070 112    .text
user_time               LABEL     0074 116    .text
wait_loop               LABEL     0052 82     .text
wait_user_clr           LABEL     0034 52     .text

Code listing
   1
   2              ; Definicao dos valores dos simbolos utilizados no programa
   3              ;
   4                    .equ  CPSR_BIT_I, 0b010000          ; Mascara para o bit I
do registo CPSR
   5
   6                    .equ  STACK_SIZE, 64                ; Dimensao do stack - 64
B
   7
```

```
   8                    ; Definicoes do porto de entrada
   9                    .equ  INPORT_ADDRESS, 0xFF80        ; Endereco do porto de
entrada
  10
  11                    ; Definicoes do porto de saida
  12                    .equ  OUTPORT_ADDRESS, 0xFFC0       ; Endereco do porto de
saida
  13
  14                    .equ  OUTPORT_INIT_VAL, 0x00        ; Valor inicial do porto
de saida
  15
  16                    ; Definicoes do circuito pTC
  17
  18                    .equ  PTC_ADDRESS,  0xFF40       ; Endereco do circuito pTC
  19
  20
  21                    .equ  PTC_TCR, 0                    ; Deslocamento do
registo TCR do pTC
  22                    .equ  PTC_TMR, 2                    ; Deslocamento do
registo TMR do pTC
  23                    .equ  PTC_TC,  4                    ; Deslocamento do
registo TC do pTC
  24                    .equ  PTC_TIR, 6                    ; Deslocamento do
registo TIR do pTC
  25
  26                    .equ  PTC_CMD_START, 0              ; Comando para iniciar a
contagem no pTC
  27                    .equ  PTC_CMD_STOP, 1               ; Comando para parar a
contagem no pTC
  28
  29
  30                    .equ  SYSCLK_FREQ, 0x09             ; Intervalo de contagem
do circuito pTC
  31                                                        ; que suporta a
implementaÃ§Ã£o do sysclk.
  32
  33
  34                    ; Outras definicoes
  35                  .equ    SYSTEM_INIT,      0xFF
  36                    .equ    USER_MASK,        0x01
  37                    .equ    TIME_MASK,        0xF0
  38                  .equ    STIMULUS_MASK,    0x01
  39                  .equ    RESULT_MASK,      0xFE
  40                  .equ    AVG_TIME,         20          ; Tempo mÃ©dio de
reaÃ§Ã£o em dezenas de ms. Definido tendo em conta ciclos de sysclk.
  41                  .equ    MIN_TIME,         0xFFC1      ; Limite inferior do
domÃ-nio de resultado
  42                  .equ    MAX_TIME,         0x0040      ; Limite superior do
domÃ-nio de resultado. Foi incrementado por 1 para poder suportar o uso da
instruÃ§ao bge.
  43                  .equ    OUT_OF_RANGE,     0xFFC0      ; Valor para quando o
resultado se encontra fora do domÃ-nio
  44                  .equ    STATE_STANDBY, 0
  45                  .equ    STATE_WAIT_STIMULUS, 1
  46                  .equ    STATE_WAIT_USER_RES, 2
  47                  .equ    RES_DISPLAY_TIME, 5           ; Tempo de display do
resultado em segundos
  48                  .equ    SLEEP_0,    50                ; Tempos de espera em
dezenas de ms.
  49                  .equ    SLEEP_1,    100
```

```
  50                    .equ    SLEEP_2,    200
  51                    .equ    SLEEP_3,    300
  52                    .equ    SLEEP_4,    400
  53                    .equ    SLEEP_5,    500
  54                    .equ    SLEEP_6,    600
  55                    .equ    SLEEP_7,    700
  56                    .equ    SLEEP_8,    800
  57                    .equ    SLEEP_9,    900
  58                    .equ    SLEEP_10,   1000
  59                    .equ    SLEEP_11,   550
  60                    .equ    SLEEP_12,   650
  61                    .equ    SLEEP_13,   750
  62                    .equ    SLEEP_14,   850
  63                    .equ    SLEEP_15,   950
  64                ; Seccao:    startup
  65                ; Descricao: Guarda o cÃ³digo de arranque do sistema
  66                ;
  67                      .section startup
  68 0000 01 58         b     _start
  69 0002 4F 0C         ldr   pc, isr_addr
  70                _start:
  71 0004 1D 0C         ldr   sp, tos_addr
  72 0006 1F 0C         ldr   pc, main_addr
  73
  74                tos_addr:
  75 0008 D6 01         .word tos
  76                main_addr:
  77 000A 0E 00         .word main
  78                isr_addr:
  79 000C EC 00         .word isr
  80
  81                ; Seccao:    text
  82                ; Descricao: Guarda o cÃ³digo do programa
  83                ;
  84                      .text
  85
  86                ; Rotina:    main
  87                ; Descricao: Inicializa o sistema, ativando o atendimento a
pedidos de interrupÃ§ao
  88                main:
  89 000E 00 60         mov   r0, #OUTPORT_INIT_VAL
  90 0010 91 5C         bl    outport_init
  91 0012 90 60         mov   r0, #SYSCLK_FREQ
  92 0014 A4 5C         bl    ptc_init
  93 0016 60 B0         mrs   r0, cpsr
  94 0018 01 61         mov   r1, #CPSR_BIT_I
  95 001A 80 C8         orr   r0, r0, r1
  96 001C 40 B0         msr   cpsr, r0
  97 001E 00 60       mov r0, #STATE_STANDBY
  98
  99                /* MÃ¡quina de estados.*/
 100                asm:
 101 0020 41 60         mov r1, #4
 102 0022 80 B8         cmp r0, r1
 103 0024 FD 4F         bcc asm
 104 0026 81 E0         lsl r1, r0, #1
 105 0028 9F 87         add pc, r1, pc
 106                states:
 107 002A 02 58         b state_standby
 108 002C 07 58         b state_wait_stimulus
```

```
 109 002E 1C 58       b state_wait_user_res
 110
 111              /* Estado standby. Coloca todos os bits RESULT e STIMULUS a 1.
Espera para user ser colocado a 1*/
 112              state_standby:
 113 0030 F0 6F       mov r0, #SYSTEM_INIT
 114 0032 71 5C       bl outport_set_bits
 115              wait_user_clr:
 116 0034 32 5C       bl  check_user
 117 0036 FE 47       bzc wait_user_clr
 118 0038 10 60       mov r0, #STATE_WAIT_STIMULUS
 119 003A F2 5B       b asm
 120
 121              /* Estado wait stimulus. Coloca os bits Result a 0 e espera o
tempo definido pelos bits TIME para baixar stimulus e proceder
 122              Ã  mediÃ§Ã£o do tempo de reaÃ§Ã£o do utilizador. Caso o utilizador
coloque user a 0 antes do tempo ser atingido, volta ao estado inicial.*/
 123              state_wait_stimulus:
 124 003C 2E 5C       bl check_user
 125 003E FE 43       bzs state_wait_stimulus         ; espera para user ser
colocado a 1
 126 0040 E0 6F       mov r0, #RESULT_MASK
 127 0042 70 5C       bl outport_clear_bits           ; limpa os bits result
 128 0044 00 6F       mov r0, #TIME_MASK
 129 0046 62 5C       bl inport_read_mask
 130 0048 3D 5C       bl get_time_value
 131 004A 38 5C       bl get_sleep_time               ; obter tempo de espera em
dezenas de ms
 132 004C 04 B0       mov r4, r0
 133 004E 4B 5C       bl  sysclk_get_ticks              ; guardar valor sysclk
inicial em r5
 134 0050 05 B0       mov r5, r0
 135              wait_loop:
 136 0052 23 5C       bl check_user                   ; caso o utilizador desative
USER, volta para o primeiro estado.
 137 0054 07 40       bzs goto_init
 138 0056 80 B2       mov r0, r5                      ; verifica se passou o tempo
timeout ou nÃ£o.
 139 0058 01 B2       mov r1, r4                      ; se ainda nÃ£o passou,
permanece em loop.
 140 005A 23 5C       bl  is_elapsed
 141 005C 00 C0       and r0, r0, r0
 142 005E F9 43       bzs wait_loop
 143              goto_wait_user_res:
 144 0060 20 60       mov r0, #STATE_WAIT_USER_RES    ; obter tempo de reacao
 145 0062 DE 5B       b asm
 146              goto_init:
 147 0064 00 60       mov r0, #STATE_STANDBY          ; voltar ao estado inicial de
espera
 148 0066 DC 5B       b asm
 149
 150
 151              /* Estado wait user response. Coloca o bit Stimulus a 0 e espera
para o utilizador reagir atravÃ©s da ativaÃ§Ã£o do sinal USER.
 152              ApÃ³s a reaÃ§Ã£o do utilizador, Ã© medido o tempo de reaÃ§Ã£o e Ã©
mostrado nos LEDS o resultado no domÃ-nio -63 a 63, em relaÃ§Ã£o ao tempo 200ms.
 153              Caso o resultado nÃ£o se encontre neste domÃ-nio, Ã© mostrado o
resultado -64.*/
 154              state_wait_user_res:
 155 0068 10 60       mov r0, #STIMULUS_MASK
```

```
 156 006A 5C 5C       bl outport_clear_bits           ; coloca stimulus a 0
 157 006C 3C 5C       bl sysclk_get_ticks
 158 006E 04 B0       mov r4, r0                      ; guardar valor inicial sysclk
 159              user_react:
 160 0070 14 5C       bl check_user                   ; esperar  pela reaÃ§Ã£o do
utilizador
 161 0072 FE 47       bzc user_react
 162              user_time:
 163 0074 38 5C       bl  sysclk_get_ticks            ; obter tempo decorrido ao
subtrair valor inicial de valor atual
 164 0076 00 8A       sub r0, r0, r4
 165 0078 41 61       mov r1, #AVG_TIME               ; verificar se resultado se
encontra no domÃ-nio
 166 007A 80 88       sub r0, r0, r1
 167 007C 01 64       mov r1, #MAX_TIME
 168 007E 80 B8       cmp r0, r1
 169 0080 04 50       bge out_of_range
 170 0082 A1 0D       ldr r1, MIN_TIME_ADDR
 171 0084 80 B8       cmp r0, r1
 172 0086 01 54       blt out_of_range
 173 0088 01 58       b show_res
 174              out_of_range:
 175 008A 70 0D       ldr r0, OUT_OF_RANGE_ADDR
 176              show_res:
 177 008C 1D 5C       bl get_res                     ; Escrever resultado nos bits
RESULT
 178 008E 43 5C       bl outport_set_bits
 179 0090 50 60       mov r0, #RES_DISPLAY_TIME
 180 0092 14 5C       bl get_sleep_time              ; Esperar tempo definido para
display de resultado.
 181 0094 1B 5C       bl delay
 182 0096 00 60       mov r0, #STATE_STANDBY          ; Voltar ao estado inicial
 183 0098 C3 5B       b asm
 184
 185              ;Verifica o estado do bit USER
 186              check_user:
 187 009A 0E 24       push lr
 188 009C 10 60       mov r0, #USER_MASK
 189 009E 36 5C       bl inport_read_mask
 190 00A0 0F 04       pop pc
 191
 192              ; Rotina:    is_elapsed
 193              ; Descricao: Verifica se ja passou o tempo definido por timeout,
tendo como referencia um tempo inicial
 194              ;            r0 = tempo inicial
 195              ;            r1 = timeout
 196              is_elapsed:
 197 00A2 0E 24       push lr
 198 00A4 00 24       push r0
 199 00A6 01 24       push r1
 200 00A8 1E 5C       bl sysclk_get_ticks
 201 00AA 02 04       pop r2
 202 00AC 01 04       pop r1
 203 00AE 80 88       sub r0, r0, r1
 204 00B0 00 B9       cmp r0, r2
 205 00B2 01 4C       bhs is_elapsed_end
 206              elapsed_return_false:
 207 00B4 00 60       mov r0, #0
 208              is_elapsed_end:
 209 00B6 0F 04       pop pc
```

```
 210
 211                MIN_TIME_ADDR:
 212 00B8 C1 FF         .word MIN_TIME
 213
 214                OUT_OF_RANGE_ADDR:
 215 00BA C0 FF         .word OUT_OF_RANGE
 216
 217                ; Rotina : get_sleep_time
 218                ; DescriÃ§Ã£o : Obtem o valor sleep pertencente ao array, com base
no indice passado em r0
 219                ; r0 = Indice do array sleep
 220                get_sleep_time:
 221 00BC 80 E0         lsl r0, r0, #1
 222 00BE 21 0E         ldr r1, sleep_ADDR
 223 00C0 10 10         ldr r0, [r1, r0]
 224 00C2 0F B7         mov pc, lr
 225
 226                ; Rotina : get_time_value
 227                ; DescriÃ§Ã£o : Obtem o valor de 1-15 definido pelos bits TIME
 228                get_time_value:
 229 00C4 00 EA         lsr r0, r0, #4
 230 00C6 0F B7         mov pc, lr
 231
 232                ; Rotina : get_res
 233                ; DescriÃ§Ã£o : Obtem o valor de resultado de 7 bits a colocar nos
LEDS.
 234                ; r0 = Valor de resultado a 8 bits.
 235                get_res:
 236 00C8 80 E0         lsl r0, r0, #1
 237 00CA 0F B7         mov pc, lr
 238
 239                ; Rotina:    delay
 240                ; Descricao: Rotina bloqueante que realiza uma espera ativa por
teste sucessivo
 241                ;            do valor da variÃ¡vel global sysclk. O tempo a
esperar, em
 242                ;            dezenas de milissegundos, e passado em R0.
 243                delay:
 244 00CC 0E 24         push lr
 245 00CE 05 24         push r5
 246 00D0 04 24         push r4
 247 00D2 04 B0         mov r4, r0
 248 00D4 08 5C         bl sysclk_get_ticks    ;guardar valor sysclk inicial em r5
 249 00D6 05 B0         mov r5, r0
 250                delay_loop:
 251 00D8 06 5C         bl sysclk_get_ticks    ;obter valor atual
 252 00DA 80 8A         sub r0, r0, r5         ;obter nÃºmero de incrementaÃ§Ãµes em
sysclk desde inicio
 253 00DC 00 BA         cmp r0, r4             ;parar se sysclk foi incrementado hms
ou mais vezes
 254 00DE FC 4B         blo delay_loop
 255                delay_end:
 256 00E0 04 04         pop r4
 257 00E2 05 04         pop r5
 258 00E4 0F 04         pop pc
 259
 260                ; Rotina:    sysclk_get_ticks
 261                ; Descricao: Devolve o valor corrente da variÃ¡vel global sysclk.
 262                ;            Interface exemplo: uint16_t sysclk_get_ticks ( );
 263                ; Entradas:  -
```

```
 264                ; Saidas:    *** Para completar ***
 265                ; Efeitos:   -
 266                sysclk_get_ticks:
 267 00E6 D0 0C           ldr r0, sysclk_ADDR
 268 00E8 00 00           ldr r0, [r0, #0]
 269 00EA 0F B7           mov pc, lr
 270
 271
 272                ; *** Inicio de troco para completar ***
 273
 274                ; *** Fim de troco para completar ***
 275
 276                ; Rotina:    isr
 277                ; Descricao: Incrementa o valor da variÃ¡vel global sysclk.
 278                ; Entradas:  -
 279                ; Saidas:    -
 280                ; Efeitos:   *** Para completar ***
 281                isr:
 282 00EC 00 24           push r0
 283 00EE 01 24           push r1
 284 00F0 F0 0F           ldr r0, PTC_ADDR
 285 00F2 00 2B           strb r0, [r0, #PTC_TIR] ; limpar o pedido
 286 00F4 60 0C           ldr r0, sysclk_ADDR     ; incrementar
 287 00F6 01 00           ldr r1, [r0, #0]
 288 00F8 91 A0           add r1, r1, #1
 289 00FA 01 20           str r1, [r0, #0]
 290 00FC 01 04           pop r1
 291 00FE 00 04           pop r0
 292 0100 20 B0           movs  pc, lr
 293
 294                sysclk_ADDR:
 295 0102 94 01           .word sysclk
 296
 297
 298                sleep_ADDR:
 299 0104 72 01           .word sleep_times
 300
 301
 302
 303                ; Gestor de perifÃ©rico para o porto de entrada
 304                ;
 305
 306                ; Rotina:    inport_read
 307                ; Descricao: Adquire e devolve o valor corrente do porto de
entrada.
 308                ;            Interface exemplo: uint8_t inport_read( );
 309                ; Entradas:  -
 310                ; Saidas:    R0 - valor adquirido do porto de entrada
 311                ; Efeitos:   -
 312                inport_read:
 313 0106 61 0C           ldr   r1, inport_addr
 314 0108 10 08           ldrb  r0, [r1, #0]
 315 010A 0F B7           mov   pc, lr
 316
 317                ; Rotina:    inport_read_mask
 318                ; Descricao: Adquire e devolve o valor corrente do porto de
entrada nos bits mask.
 319                ;            Interface exemplo: uint8_t inport_read( uint8_t mask
);
 320                ; Entradas:  R0 - valor mask
```

```
 321                ; Saidas:    R0 - valor adquirido do porto de entrada nos bits
mask
 322                ; Efeitos:   -
 323                inport_read_mask:
 324 010C 31 0C           ldr r1, inport_addr
 325 010E 11 08           ldrb r1, [r1, #0]
 326 0110 80 C0           and r0, r0, r1
 327 0112 0F B7           mov pc, lr
 328
 329                inport_addr:
 330 0114 80 FF           .word INPORT_ADDRESS
 331
 332
 333
 334                ; Gestor de perifÃ©rico para o porto de saÃ-da
 335                ;
 336
 337                ; Rotina:    outport_set_bits
 338                ; Descricao: Atribui o valor logico 1 aos bits do porto de saida
identificados
 339                ;            com o valor 1 em R0. O valor dos outros bits nao e
alterado.
 340                ;            Interface exemplo: void outport_set_bits( uint8_t
pins_mask );
 341                ; Entradas:  R0 - Mascara com a especificacao do indice dos bits a
alterar.
 342                ; Saidas:    -
 343                ; Efeitos:   Altera o valor da variavel global outport_img.
 344                outport_set_bits:
 345 0116 0E 24           push  lr
 346 0118 21 0D           ldr   r1, outport_img_addr
 347 011A 12 08           ldrb  r2, [r1, #0]
 348 011C 20 C8           orr   r0, r2, r0
 349 011E 10 28           strb  r0, [r1, #0]
 350 0120 0F 5C           bl    outport_write
 351 0122 0F 04           pop   pc
 352
 353                ; Rotina:    outport_clear_bits
 354                ; Descricao: Atribui o valor logico 0 aos bits do porto de saida
identificados
 355                ;            com o valor 1 em R0. O valor dos outros bits nao e
alterado.
 356                ;            Interface exemplo: void outport_clear_bits( uint8_t
pins_mask );
 357                ; Entradas:  R0 - Mascara com a especificacao do indice dos bits a
alterar.
 358                ; Saidas:    -
 359                ; Efeitos:   Altera o valor da variavel global outport_img.
 360                outport_clear_bits:
 361 0124 0E 24           push  lr
 362 0126 B1 0C           ldr   r1, outport_img_addr
 363 0128 12 08           ldrb  r2, [r1, #0]
 364 012A 10 B0           mvn   r0, r0
 365 012C 20 C0           and   r0, r2, r0
 366 012E 10 28           strb  r0, [r1]
 367 0130 07 5C           bl    outport_write
 368 0132 0F 04           pop   pc
 369
 370                ; Rotina:    outport_init
```

```
 371                ; Descricao: Faz a iniciacao do porto de saida, nele estabelecendo
o valor
 372                ;            recebido em R0.
 373                ;            Interface exemplo: void outport_init( uint8_t value
);
 374                ; Entradas:  R0 - Valor a atribuir ao porto de saida.
 375                ; Saidas:    -
 376                ; Efeitos:   Altera o valor da variavel global outport_img.
 377                outport_init:
 378 0134 0E 24           push  lr
 379 0136 31 0C           ldr   r1, outport_img_addr
 380 0138 10 28           strb  r0, [r1]
 381 013A 02 5C           bl    outport_write
 382 013C 0F 04           pop   pc
 383
 384                outport_img_addr:
 385 013E 92 01           .word outport_img
 386
 387                ; Rotina:    outport_write
 388                ; Descricao: Escreve no porto de saida o valor recebido em R0.
 389                ;            Interface exemplo: void outport_write( uint8_t value
);
 390                ; Entradas:  R0 - valor a atribuir ao porto de saida.
 391                ; Saidas:    -
 392                ; Efeitos:   -
 393                outport_write:
 394 0140 21 0C           ldr   r1, outport_addr
 395 0142 10 28           strb  r0, [r1, #0]
 396 0144 0F B7           mov   pc, lr
 397
 398                outport_addr:
 399 0146 C0 FF           .word OUTPORT_ADDRESS
 400
 401                ; Gestor de perifÃ©rico para o Pico Timer/Counter (pTC)
 402                ;
 403
 404                ; Rotina:    ptc_start
 405                ; Descricao: Habilita a contagem no periferico pTC.
 406                ;            Interface exemplo: void ptc_start( );
 407                ; Entradas:  -
 408                ; Saidas:    -
 409                ; Efeitos:   -
 410                ptc_start:
 411 0148 30 0D           ldr   r0, PTC_ADDR
 412 014A 01 60           mov   r1, #PTC_CMD_START
 413 014C 01 28           strb  r1, [r0, #PTC_TCR]
 414 014E 0F B7           mov   pc, lr
 415
 416
 417
 418                ; Rotina:    ptc_stop
 419                ; Descricao: Para a contagem no periferico pTC.
 420                ;            Interface exemplo: void ptc_stop( );
 421                ; Entradas:  -
 422                ; Saidas:    -
 423                ; Efeitos:   O valor do registo TC do periferico e colocado a
zero.
 424                ptc_stop:
 425 0150 F0 0C           ldr   r0, PTC_ADDR
 426 0152 11 60           mov   r1, #PTC_CMD_STOP
```

```
 427 0154 01 28          strb  r1, [r0, #PTC_TCR]
 428 0156 0F B7          mov   pc, lr
 429
 430
 431                ; Rotina:    ptc_get_value
 432                ; Descricao: Devolve o valor corrente da contagem do periferico
pTC.
 433                ;            Interface exemplo: uint8_t ptc_get_value( );
 434                ; Entradas:  -
 435                ; Saidas:    R0 - O valor corrente do registo TC do periferico.
 436                ; Efeitos:   -
 437                ptc_get_value:
 438 0158 B1 0C          ldr   r1, PTC_ADDR
 439 015A 10 0A          ldrb r0, [r1, #PTC_TC]
 440 015C 0F B7          mov   pc, lr
 441
 442                ; Rotina:    ptc_init
 443                ; Descricao: Inicia uma nova contagem no periferico pTC com o
intervalo de
 444                ;            contagem recebido em R0, em ticks.
 445                ;            Interface exemplo: void ptc_init( uint8_t interval );
 446                ; Entradas:  R0 - Valor do novo intervalo de contagem, em ticks.
 447                ; Saidas:    -
 448                ; Efeitos:   Inicia a contagem no periferico a partir do valor
zero, limpando
 449                ;            o pedido de interrupcao eventualmente pendente.
 450                ptc_init:
 451 015E 0E 24       push    lr
 452 0160 00 24       push    r0
 453 0162 F6 5F       bl        ptc_stop
 454 0164 00 04       pop         r0
 455 0166 41 0C       ldr         r1, PTC_ADDR
 456 0168 10 29       strb    r0, [r1, #PTC_TMR]
 457 016A 11 2B       strb    r1, [r1, #PTC_TIR]
 458 016C ED 5F         bl ptc_start
 459 016E 0F 04       pop         pc
 460
 461
 462                PTC_ADDR:
 463 0170 40 FF          .word PTC_ADDRESS
 464
 465                ; Seccao:    data
 466                ; Descricao: Guarda as variÃ¡veis globais com um valor inicial
definido
 467                ;
 468                      .data
 469
 470                ; array com os tempos sleep do sistema.
 471                sleep_times:
 472                     .word SLEEP_0, SLEEP_1, SLEEP_2, SLEEP_3, SLEEP_4, SLEEP_5,
SLEEP_6, SLEEP_7, SLEEP_8, SLEEP_9, SLEEP_10, SLEEP_11, SLEEP_12, SLEEP_13,
SLEEP_14, SLEEP_15
 472 0172 32 00 64 00 C8 00 2C 01 90 01 F4 01 58 02 BC 02
 472 0182 20 03 84 03 E8 03 26 02 8A 02 EE 02 52 03 B6 03
 473
 474                ; Seccao:    bss
 475                ; Descricao: Guarda as variÃ¡veis globais sem valor inicial
definido
 476                ;
 477                      .bss
```

```
 478                    outport_img:
 479 0192 00                   .space      1
 480 0193 00                   .align
 481
 482                    sysclk:
 483 0194 00                   .space      2
 483 0195 00
 484                    ; Seccao:    stack
 485                    ; Descricao: Implementa a pilha com o tamanho definido pelo
simbolo STACK_SIZE
 486                    ;
 487                           .stack
 488 0196 00                   .space      STACK_SIZE
 488 .... ..
 488 01D5 00
 489                    tos:
```